Lab 1

Matlab Fundamentals; Part I

"Learning to walk"

Handout – print version of Lecture on *Marine Modelling* January 14, 2019

Katja Fennel, Oceanography, Dalhousie University

1.1

1 Introduction

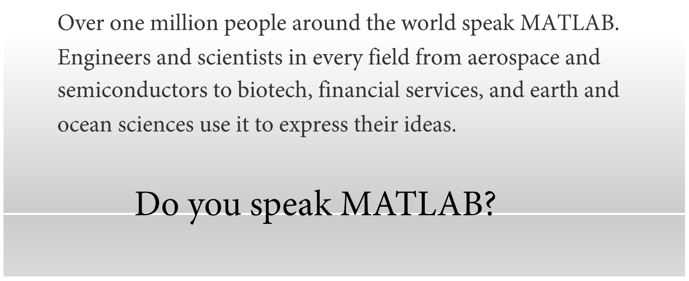
Marine Modelling without ever touching a computer would be a bit like theoretical bicycle riding. There is no better way to learn than with hands on experience, in this case, hands on a keyboard.

MATLAB is a widely used, very powerful tool for **data analysis, complex calculations** and **visualization**; of course there are others. One of the features that distinguish Matlab from others such as FORTRAN, C++ or Java (and make it particularly well suited as a learning tool and a gateway to programming in general) is that it can be used **interactively**. This means one can type commands at the Matlab prompt, hit enter and get the answer immediately.

Introduction

Advantages of MATLAB:

- powerful and widely used,
- useful for data analysis, complex calculations and visualization,
- works interactively (unlike FORTRAN, C++ or Java),
- customizable.



Over one million people around the world speak MATLAB. Engineers and scientists in every field from aerospace and semiconductors to biotech, financial services, and earth and ocean sciences use it to express their ideas.

Do you speak MATLAB?

1.2

Two essential requirements for successful programming (in Matlab or any other computer language):

- one needs to know the exact rules and syntax for writing statements (computers are stupid, they will do exactly as told);
- one needs to develop a logical plan for solving the problem under consideration.

In the first few labs we will focus on dealing with the first requirement and get familiar with basic Matlab rules and syntax. Next we'll move on to dealing with more complex problems and ways of solving them (the second point).

At the end of the course you will (hopefully) have an appreciation for how Matlab can help you in designing, developing an implementing computational and graphical tools for your scientific problems. For example, you will be able to develop a personal toolbox containing specialized code for your specific tasks.

Introduction

Two essential requirements for successful programming:

- one needs to know the exact rules and syntax for writing statements (computers are stupid, they will do exactly as told);
- one needs to develop a logical plan for solving the problem under consideration.

This and next lab: we will focus on the first requirement; basic Matlab rules and syntax

Next we'll deal with more complex problems and ways of solving them (the second point).

At the end of the course you will be able to use MATLAB in designing, developing an implementing computational and graphical tools for your scientific problems (e.g. personal toolbox tailored to your scientific problems). 1.3

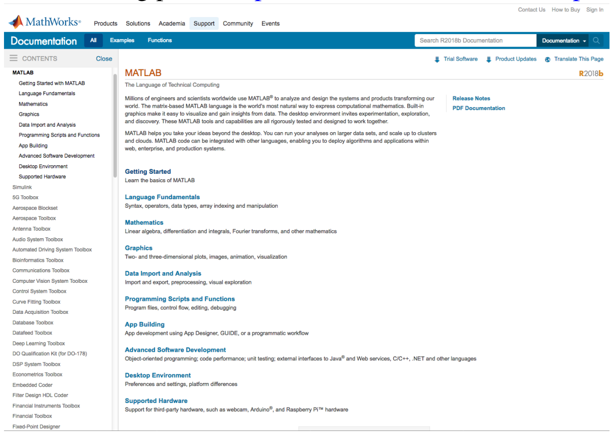
There are several places where you can get help. The most easily accessible is maybe http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html (you can find a printable documentation at the site as well as demos, tutorials and other resources).

Introduction

Getting help:

- excellent built-in help,
- lots of online resources (demos, tutorials etc.)

Good starting point: https://www.mathworks.com/help/matlab/index.html



MATLAB R2018b

The Language of Technical Computing

Millions of engineers and scientists worldwide use MATLAB® to analyze and design the systems and products transforming our world. The matrix-based MATLAB language is the world's most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

MATLAB helps you take your ideas beyond the desktop. You can run your analyses on larger data sets, and scale up to clusters and clouds. MATLAB code can be integrated with other languages, enabling you to deploy algorithms and applications within web, enterprise, and production systems.

Getting Started
Learn the basics of MATLAB

Language Fundamentals
Syntax, operators, data types, array indexing and manipulation

Mathematics
Linear algebra, differentiation and integrals, Fourier transforms, and other mathematics

Graphics
Two- and three-dimensional plots, images, animation, visualization

Data Import and Analysis
Import and export, preprocessing, visual exploration

Programming Scripts and Functions
Program files, control flow, editing, debugging

App Building
App development using App Designer, GUIDE, or a programmatic workflow

Advanced Software Development
Object-oriented programming; code performance; unit testing; external interfaces to Java® and Web services, C/C++, .NET and other languages

Desktop Environment
Preferences and settings, platform differences

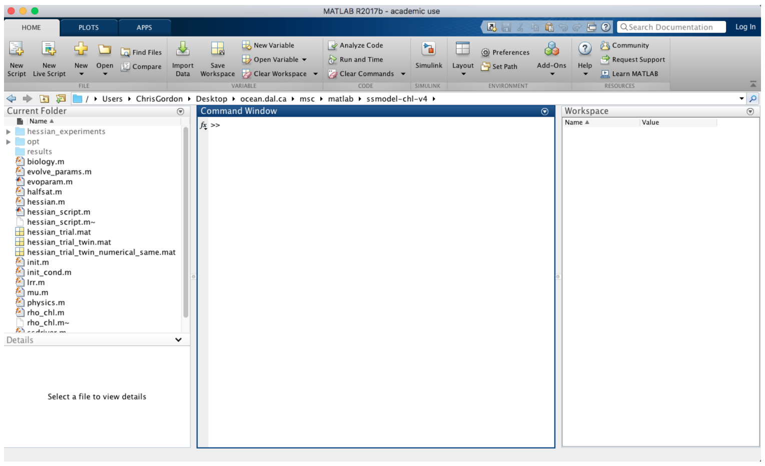
Supported Hardware
Support for third-party hardware, such as webcam, Arduino®, and Raspberry Pi™ hardware

1.4

2 Fundamentals

Fundamentals

Start by double-clicking the Matlab icon or typing matlab at the prompt.



1.5

For now we will only work in the Command Window.

Fundamentals

To *exit* either select **Exit Matlab** from the **File** menu, or

Enter `quit` or `exit` at the Command Window prompt (»).

Note: Do not click the close box (cross in the top corner). 1.6

Matlab as a calculator

Enter the following commands:

```
>> 2+3
>> 2-3
>> 2*3
>> 1/2
>> 2^3
>> 2\1
```

What do the symbols

```
*,/,^,\
```

mean?

Try this:

```
>> 2 .* 3
>> 1 ./ 2
>> 2 .^ 3
```

1.7

In these examples the dot doesn't change the result because the operations are done with scalars (single numbers). We'll see later how the dots affect the result when dealing with matrices.

A few things to note:

- You can edit the Matlab command using the **Backspace**, **Left-arrow**, **Right-arrow** and **Del** keys.
- You can make previously entered commands reappear using the **Up-arrow** and **Down-arrow** commands (typing enter will execute the command).
- *Smart recall* feature: Type the first few characters of a previously entered command and press the Up-arrow key.

How will Matlab handle $0/1$ or $1/0$? Try it! 1.8

Matlab tries to anticipate errors and will warn you. *Inf* is the Matlab symbol for infinity. If you want to use infinity in a calculation you can.

How about 0/0?

NaN stands for not-a-number. Can be useful when working with data fields which contain missing values (will become more obvious later). 1.9

Assigning values to variables

```
>> a = 2
>> a = a + 2

>> b = 3;
>> c = a + b;
>> c

>> x = 2; y = 3; z = x + y;
>> z

>> A
>> B
```

Note:

- The semicolon suppresses display and allows several commands per line.
- MATLAB is *case-sensitive*.

1.10

Note how the semicolon suppresses the display of the values of b and c. Also note how it allows several commands to be entered on one line.

Also note that MATLAB is *case sensitive*; a variable named A is not the same as a. Nor is Alpha the same as ALPHA or alpha or AlPhA. They are all different. If you don't want MATLAB to regurgitate all the numbers that are an answer to the statement you just entered, be sure to finish your command with a semicolon (;).

Matlab has the usual mathematical functions that you would expect to find on a calculator (pi, sin, cos, log).

- Find $\sqrt{\pi}$ as sqrt(pi) (should be 1.7725)
- Trigonometric functions like sin(x) expect the argument in radians (multiply degrees by $\frac{\pi}{180^\circ}$ to get radians). Try *sin(90°)* as sin(90*pi/180) (should be 1)
- The exponential function e^x is expressed as exp(x). Use this information to find the values of e and $\frac{1}{e}$ (should be 2.7183 and 0.3679)

Note: Be careful when naming your own functions. 1.11

Note: Because of the numerous built-in functions (pi, sin etc.) one should be careful not to name ones own variables the same (unless there is a good reason). An illustration follows.

Illustration of a naming conflict

Try:

```
>> pi = 4;
>> sqrt(pi)
>> whos
>> clear pi
>> whos
>> sqrt(pi)
>> clear
>> whos
```

Note: `clear` deletes all variables from the workspace. `clear pi` only deletes pi. `whos` shows all local variables in the workspace. 1.12

Matlab has many general functions. For example, try

```
>> date
>> calendar
```

1.13

Matlab has many commands for a variety of tasks. For example, `clc`, `help` and `lookfor`. If you don't know the name of a command, but, for example, want to know how to make an identity matrix, type `lookfor identity`. You will then be told about `eye`. Use `help eye` to find out about the eye command. Note that MATLAB capitalizes things in its help messages for EMPHASIS, which confuses things a little. Commands and functions are always in lower case, although they are capitalized in the help messages.

Matlab has many commands for a variety of tasks, e.g. `clc`, `help` and `lookfor`. For example,

```
>> clc      %  clears command window
>> help clc
>> lookfor identity
```

1.14

Matlab can handle vectors and matrices (generally referred to as arrays). An easy way to define a vector that has a constant increment between its elements is thus:

```
>> x = 1 : 10;
```

Then check:

```
>> x
```

x is a row vector with 11 elements (columns). Check it size thus:

```
>> size(x)
```

Try

```
>> y = 2 * x
>> w = y ./ x
>> z = sin(x)
```

Note that the 2nd line is an array operation; the division is carried out element-by-element. 1.15

Drawing a graph of sin(x):

```
>> x = 0 : 0.1 :2*pi;
>> z = sin(x);
>> plot(x,z), grid
```

Note that the first command has three numbers and two semicolons. In this case the middle number specifies the increment (default is 1). The command grid adds grid lines to the graph.

You can add more graphs, axis labels, a title and a legend to the current figure:

```
>> hold on
>> plot(x,cos(x),'r')
>> xlabel('x')
>> ylabel('y')
>> title('sin and cos')
>> legend('sin','cos')
```

1.16 You can clear the current figure window using `clf` or open a new one using `figure`.

You can also use different line styles (dashed, dotted, different symbols).

1.17 Use `help plot` to see the different options and make a new plot using different line styles.
Back to arrays. Vectors and matrices can also be entered element-by-element.

```
>> V = [1 2 3]
>> M = [1 2 3; 4 5 6; 7 8 9]
```

And one can refer to the individual elements:

```
>> M(1,1)
>> M(2,3)
```

And overwrite elements:

```
>> M(2,3) = 10
```

And concatenate arrays (if their dimensions allow):

```
>> M = [M; V]
```

1.18 The colon operator (:) is a useful thing to keep in mind and learn more about. For example, `j:k` is equivalent to `[j, j+1, j+2, ..., k]` or j:d:k is equivalent to [j, j+d, j+2*d, ..., j+m*d] where `m = fix((K- J)/D)`. The colon operator can be used to pick out specific rows, columns, elements of arrays. Learn more with help colon.
The colon operator is used to refer to rows or columns or subsets of rows and columns:

```
>> M(:,1)
>> M(2:3,2:3)
```

The prime operator turns rows into columns or vice versa (also called transpose):

```
>> W = V'
```

Built-in functions exist for some frequently used matrices:

```
>> E = ones(4,3)
>> Z = zeros(4,3)
```

1.19 Try the following. Does Matlab behave like you would expect? Can you explain the behavior you see?

```
>> M + E
>> M + E'
>> M + 4
>> M * 2
>> M * E
```

1.20 To check the size of arrays:

```
>> size(M)
>> size(V)
>> length(V)
```

Note the two different ways of multiplying matrices in Matlab:

```
>> M * ones(3,4) % matrix multiplication
>> M .* M  % multiplication element-by-element
% or array multiplication
```

Remember that matrix multiplication is not the same as scalar or array multiplication; the latter is designated with a "dot" before it. For example `C=A*B` is a matrix multiplication, whereas `C=A.*B` is array multiplication. In the case of matrix multiplication the elements of C are the scalar product of the corresponding rows of A and columns of B. In the latter case the operation is done element by element. The array dimensions have to match according to the operation performed. 1.21

Note the difference between:

```
>> who
>> whos
```

Another useful feature is tab completion. 1.22

Type the first few letters of a matlab command and then the tab key. If the name is unique, Matlab will complete it for you. Otherwise, type tab gain and a list of possible completions appears.

For a sample of Matlab's features, try `demo` at the command line. Especially check out the different graphics demos. 1.23